

SVI 画像入力ライブラリ説明書

V1. 32

株式会社ネットビジョン

改定履歴

版数	日付	内容	備考
1. 00	2011/11/02	<ul style="list-style-type: none"> SVI-03 用画像入力ライブラリ V1.05 より引用 SVI-06 用に 3.4.31. SVIUSB20_Fx3FirmUpdate、3.4.32. SVIUSB20_ReconFpgaUpdate を追加しました。 	
1. 01	2012/08/23	<ul style="list-style-type: none"> 誤字脱字修正 	
1. 02	2013/02/01	<ul style="list-style-type: none"> 3.4.38. SVI-06 の設定で I2C API の使用法に間違いがありました。 誤字脱字修正 	
1. 03	2013/09/12	<ul style="list-style-type: none"> 3.4.38. SVI-06 の設定で例 4 として VSYNC 反転制御を追記しました。 	
1. 04	2014/01/26	<ul style="list-style-type: none"> 3.4.28. SVIUSB20_DeviceSelect の引数で SVI-06 のボード番号を 0-3 から 0-7 に変更しました。 3.4.37. SVIUSB20_DeviceSelect の引数で SVI-06 のボード番号を 0-3 から 0-7 に変更しました。 	
1. 10	2019/03/18	<ul style="list-style-type: none"> SVI-09 サポートにつき「SVI-06 画像入力ライブラリ説明書」から「SVI 画像入力ライブラリ説明書」に文書名を変更 	
1. 20	2019/8/8	<ul style="list-style-type: none"> 対象に SVM-06 ボードを追加 SVIUSB20_RecStartEX API を追加 SVIUSB20_RecGetData API を追加 SVIUSB20_GetStatus API の構造体を変更 	
1. 30	2022/03/09	<ul style="list-style-type: none"> SVIUSB20_RecGetHeader API の構造体のメンバ設定範囲を変更 SVIUSB20_RecGetFrames API の 24/32bit レコーディング時のデータ格納フォーマットを変更 	
1. 31	2022/12/26	<ul style="list-style-type: none"> 販売終了により SVI-06 ボードの情報を削除 SVP-01-V ボードをサポート対象として追加 3.4.25. SVIUSB20_RecStart2 API を追加 3.4.26. SVIUSB20_RecDataRead API を追加 3.4.27. SVIUSB20_RecStop2 API を追加 3.4.30. 開始トリガ無しレコーディング時…を追加 	
1. 32	2023/1/19	<ul style="list-style-type: none"> 3.4.25. SVIUSB20_RecStart2 API の説明を修正 3.4.26. SVIUSB20_RecDataRead API の説明を修正 3.4.30. 開始トリガ無しレコーディング時…を修正 3.4.31. JPEG など水平画素は一定でない場合のモニタリングの画像入力ライブラリ使用例を追加 	

目次

1. 適用.....	4
2. 概要.....	4
3. 仕様.....	5
3.1. ファイル構成 (32BITOS).....	5
3.2. ファイル構成 (64BIT OS).....	5
3.3. API一覧	6
3.4. SVI画像入力ライブラリAPIリファレンス.....	7
3.4.1. SVIUSB20_Init	7
3.4.2. SVIUSB20_Open	7
3.4.3. SVIUSB20_Close.....	8
3.4.4. SVIUSB20_GetStatus	9
3.4.5. SVIUSB20_GetStatus2	10
3.4.6. SVIUSB20_SetParam.....	11
3.4.7. SVIUSB20_MonStart	12
3.4.8. SVIUSB20_MonStop	12
3.4.9. SVIUSB20_MonGetHeader.....	13
3.4.10. SVIUSB20_MonGetFrame	14
3.4.11. SVIUSB20_RecStart.....	14
3.4.12. SVIUSB20_RecStop	15
3.4.13. SVIUSB20_RecGetHeader.....	15
3.4.14. SVIUSB20_RecGetFrames	16
3.4.15. SVIUSB20_I2COneBlockWrite.....	19
3.4.16. SVIUSB20_I2COneBlockRead.....	20
3.4.17. SVIUSB20_GpioAcc	21
3.4.18. SVIUSB20_GetVersion	21
3.4.19. SVIUSB20_End	22
3.4.20. SVIUSB20_DeviceSelect	22
3.4.21. SVIUSB20_DeviceRelease	22
3.4.22. SVIUSB20_EnumDevice.....	23
3.4.23. SVIUSB20_RecStartEx	23
3.4.24. SVIUSB20_RecGetData	24
3.4.25. SVIUSB20_RecStart2	28
3.4.26. SVIUSB20_RecGetFrames2.....	28
3.4.27. SVIUSB20_RecStop2.....	31

3.4.28.	モニタリングモード時の画像入力ライブラリ使用例.....	32
3.4.29.	レコーディングモード時の画像入力ライブラリ使用例	34
3.4.30.	開始トリガなしレコーディング時の画像入力ライブラリ使用例	36
3.4.31.	JPEGなど水平画素は一定でない場合のモニタリングの画像入力ライブラリ使用例	38
3.4.32.	I2Cによるコマンド送信時の画像入力ライブラリ使用例	40
3.4.33.	I2Cによるコマンド受信時の画像入力ライブラリ使用例	41
3.4.34.	複数台のSVボードを使用する場合のオープン／クローズ方法	42
3.4.35.	SVボードの設定	43

1. 適用

本説明書は SVI-09/SVP-01-V/SVM-06 の各ベンダーモードに適用します。

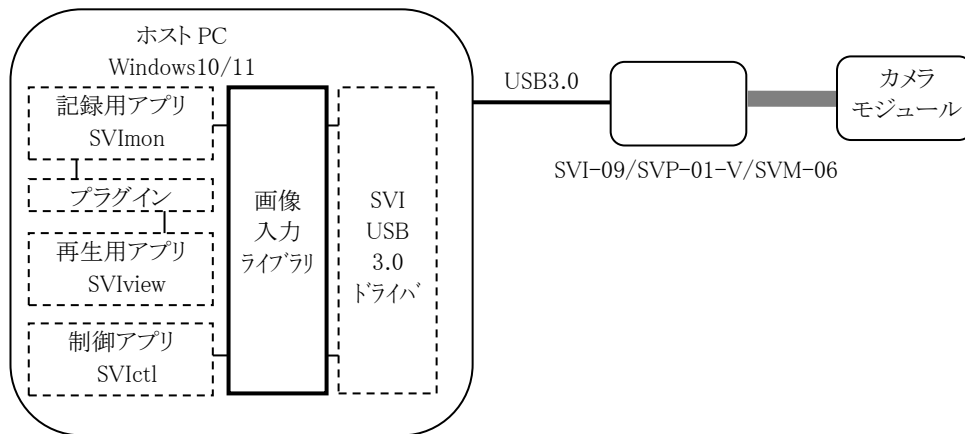
2. 概要

SVI-09、SVP-01-V、SVM-06 (以降 SV ボード) とはカメラ・モジュールを評価する Windows 上のソフトウェアとハードウェア及びファームウェアから構成されます。

本説明書では SV ボードを USB3.0 を介して接続、制御するための画像入力ライブラリについて記述します。各 SV ボードはベンダーモードで起動していることが前提となります。

画像入力ライブラリを使用することで、カメラモジュールからのモニタリング画像、レコーディング画像を取り込むことができます。また、カメラモジュール制御として I²C 通信をサポートします。

【図1】 システム構成図



3. 仕様

本ライブラリは SV ボード 専用デバイスドライバを呼び出し SV ボードより画像を入力するためのライブラリです。アプリケーションからは本ライブラリを使用して制御します。本ライブラリ内 API をコールすることにより SV ボードのパラメータの設定、ステータスの取得及び画像データの受信を実現します。

アプリケーションでは、本ライブラリ内 API を呼び出し、I²C 通信でカメラモジュールの制御や汎用ポートの制御を行うことができます。また従来通り、32bitOS 用、64bitOS 用でも異なります。

3.1. ファイル構成 (32bitOS)

本ライブラリは以下のファイルを提供します。

- SVIUSB30.INF (Software-CD の Driver フォルダに格納)
SVI USB3.0 ドライバインストール情報ファイル。
- SVIUSB30.SYS (Software-CD の Driver フォルダに格納)
SVI USB3.0 ドライバ本体 Windows ディレクトリ下”System32¥drivers”にコピーされ使用されます。
- SVIUSB20.H (Software-CD の“SVI-source¥画像入力ライブラリ¥x86”フォルダに格納)
本ライブラリを使用する際に必要なインクルードファイルです。
- SVIUSB20.DLL (Software-CD の“Appl”フォルダに格納)
本ライブラリです。
- SVIUSB20.LIB (Software-CD の“SVI-source¥画像入力ライブラリ¥x86”フォルダに格納)
本ライブラリリンクモジュールです。

3.2. ファイル構成 (64bit OS)

本ライブラリは以下のファイルを提供します。

- SviU3drv.inf (Software-CD の Driver_x64 フォルダに格納)
SVI USB3.0 ドライバインストール情報ファイル。
- SviU3drv.dll (Software-CD の Driver_x64 フォルダに格納)
SVI USB3.0 ドライバ本体 Windows ディレクトリ下”System32¥drivers”にコピーされ使用されます。
- SVIUSB20.H (Software-CD の“SVI-source¥画像入力ライブラリ¥x64”フォルダに格納)
本ライブラリを使用する際に必要なインクルードファイルです。
- SVIUSB20.DLL (Software-CD の“Appl_x64”フォルダに格納)
本ライブラリです。
- SVIUSB20.LIB (Software-CD の“SVI-source¥画像入力ライブラリ¥x64”フォルダに格納)
本ライブラリリンクモジュールです。

3.3. API 一覧

API 名	機能
SVIUSB20_Init	SVI 画像入力ライブラリを初期化します
SVIUSB20_Open	SVI ドライバをオープンします
SVIUSB20_Close	SVI ドライバをクローズします
SVIUSB20_GetStatus	SVI の現在のステータスを取得します
SVIUSB20_GetStatus2	SVI の現在の基本ステータスのみを取得します
SVIUSB20_SetParam	SVI にハードウェア設定情報を通知します
SVIUSB20_MonStart	モニタリングを開始します
SVIUSB20_MonStop	モニタリングを停止します
SVIUSB20_MonGetHeader	モニタリング中、フレームヘッダを取得します
SVIUSB20_MonGetFrame	モニタリング中、フレームデータを受信します
SVIUSB20_RecStart	レコーディングを開始します
SVIUSB20_RecStop	レコーディングを停止します
SVIUSB20_RecGetHeader	レコーディング中、ヘッダを取得します
SVIUSB20_RecGetFrames	レコーディング中、全フレームデータを受信します
SVIUSB20_I2COneBlockWrite	I2C で 1 ブロックを送信します
SVIUSB20_I2COneBlockRead	I2C で 1 ブロックを受信します
SVIUSB20_GpioAcc	SVI ボードの GPIO ポートをアクセスします
SVIUSB20_GetVersion	SVI 画像入力ライブラリ のバージョン情報を取得します
SVIUSB20_End	SVI 画像入力ライブラリの終了処理を行います
SVIUSB20_DeviceSelect	使用する SVI ボードを指定します
SVIUSB20_DeviceRelease	使用指定した SVI ボードを開放します
SVIUSB20_EnumDevice	SVI ボードの接続台数及び接続された SVI ボードの番号を接続台数分取得します
SVIUSB20_RecStartEx	リアルタイム版レコーディングを開始します (CAN 対応) ※1
SVIUSB20_RecGetData	リアルタイムにレコーディングデータを受信します (CAN 対応) ※1
SVIUSB20_RecStart2	開始トリガ無しでレコーディングを開始します ※2
SVIUSB20_RecGetFrames2	開始トリガ無しレコーディングデータを受信します ※2
SVIUSB20_RecStop2	開始トリガ無しレコーディングを停止します ※2

※1 は CAN モードで起動した場合有効です。

※2 は SVM-06 で未サポートです。

3.4. SVI 画像入力ライブラリ API リファレンス

3.4.1. SVIUSB20_Init

API SVIUSB20_Init

機能 SVI 画像入力ライブラリの内部変数を初期化します

プロトタイプ

```
void SVIUSB20_Init( void );
```

戻り値

なし

備考

・必ず最初に呼び出して下さい。(SVIUSB20_Open API よりも！)

3.4.2. SVIUSB20_Open

API SVIUSB20_Open

機能 SVI 専用デバイスドライバをオープンします

プロトタイプ

```
DWORD SVIUSB20_Open (
    ULONG          ulAppWho,          // オープン元のアプリケーションを指定
                                     // SVIUSB20_APP_WHO_REC (表示アプリ用)
                                     // SVIUSB20_APP_WHO_CTL (制御アプリ用)
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_DEVOPEN	デバイスドライバをオープンできません
SVIUSB20_RET_ERROR_MULTIOOPEN	同じアプリケーションからは 2 重にオープンできません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

3.4.3. SVIUSB20_Close

API SVIUSB20_Close

機能 SVI 専用デバイスドライバをクローズします

プロトタイプ

```
DWORD SVIUSB20_Close (  
    ULONG ulAppWho                // オープン元のアプリケーションを指定  
                                //   SVIUSB20_APP_WHO_REC (表示アプリ用)  
                                //   SVIUSB20_APP_WHO_CTL (制御アプリ用)  
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります

備考

表示アプリ用でオープンした場合は、表示アプリ用でクローズしてください

3.4.4. SVIUSB20_GetStatus

API SVIUSB20_GetStatus

機能 SV ボードの現在のステータス情報を取得します

プロトタイプ

```
DWORD SVIUSB20_GetStatus (
    PGET_STATUS pStatus // ステータス情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef _GET_STATUS {
    ULONG ulBasicStatus; // SV ボードの基本ステータス(詳細は備考参照)
    ULONG ulHWVersion; // ハードウェアのバージョン番号
    ULONG ulFWVersion; // ファームウェアのバージョン番号
    ULONG ulOPStatus; // 現在の動作モード
                        // 0:アイドル、1:モニタリング中
                        // 2:レコーディング中
    ULONG ulStartupStatus; // ファームウェア起動ステータス(0:ノーマル、1:リカバリ)
    ULONG ulOrgSizeW; // カメラが出力している画像サイズの幅
    ULONG ulOrgSizeH; // カメラが出力している画像サイズの高さ
    ULONG ulCutout_x; // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG ulCutout_y; // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG ulCutSizeW; // 切り出し後の画像サイズの幅
    ULONG ulCutSizeH; // 切り出し後の画像サイズの高さ
    ULONG ulCamStatus; // カメラステータス(未使用)
    ULONG ulCamFps; // カメラスの出力フレームレート(100 倍された値)
} GET_STATUS, *PGET_STATUS;
```

備考

実行結果として SV ボードの基本ステータスのみを取得したい場合は SVIUSB20_GetStatus2 API を使用して下さい。
SV ボードの基本ステータスの一覧を以下に示します。

SVL_STS_SUCCESS	: 正常終了
SVL_STS_FRAMEPENDING	: 正常終了 (保留フレーム/データあり)
SVL_STS_CAPCANCELED	: 正常終了 (キャプチャが中止された)
SVL_STS_BUSY	: ビジーでコマンドが実行できない
SVL_STS_RECOVERYMODE	: リカバリーモードのためコマンドが実行できない
SVL_STS_I2CACTIVE	: I2C コントローラアクティブ
SVL_STS_CMD_INVALID	: コマンドが不正である
SVL_STS_PRM_INVALID	: パラメータが不正である
SVL_STS_SEQ_INVALID	: パケットの発行シーケンスが不正である
SVL_STS_I2C_ACKTIMEOUT	: I2C でスレーブからの ACK を受信できずタイムアウトが発生した
SVL_STS_I2C_PRETIMEOUT	: I2C でプリタイムアウトが発生した
SVL_STS_I2C_POSTTIMEOUT	: I2C でポストタイムアウトが発生した
SVL_STS_USB_ERROR	: USB 通信時にエラーが発生した
SVL_STS_UPDATE_INVALID	: モジュールデータが不正である
SVL_STS_FROMERS_ERROR	: フラッシュメモリーの消去に失敗した
SVL_STS_FROMWT_ERROR	: フラッシュメモリーの書き込みに失敗した
SVL_STS_INTERNAL_ERROR	: 内部エラーが発生した
SVL_STS_RESOURCE_ERROR	: 内部リソースが不足して処理が実行できない

3.4.5. SVIUSB20_GetStatus2

API SVIUSB20_GetStatus2

機能 SV ボードの現在の基本ステータスのみを取得します

プロトタイプ

```
DWORD SVIUSB20_GetStatus2 (
    PULONG    pStatus    // 基本ステータス格納ポインタ (詳細は SVIUSB20_GetStatus と同様です)
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

3.4.6. SVIUSB20_SetParam

API SVIUSB20_SetParam

機能 SV ボードのハードウェア設定を行います

プロトタイプ

```
DWORD SVIUSB20_SetParam (
    PSET_PARAM    pParam           // ハードウェア設定構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _SET_PARAM {
    ULONG    ulParamBitFlag;           // 以下のパラメータの有効(1)/無効(0)を指定する
                                         // SVIUSB20_SETPARAM_BIT0 : 画像情報通知
                                         // SVIUSB20_SETPARAM_BIT1 : カメラ電源スイッチ
                                         // SVIUSB20_SETPARAM_BIT2 : I2C スピード
                                         // SVIUSB20_SETPARAM_BIT3 : 予約
                                         // SVIUSB20_SETPARAM_BIT4 : 予約
                                         // SVIUSB20_SETPARAM_BIT5 : VSYNC デアサートフラグ
                                         // SVIUSB20_SETPARAM_BIT6 : モニタリングモード
                                         // 複数指定は OR 演算子を使用
    ULONG    ulVsyncFlag;             // VSYNC デアサートフラグ ~ 初期値:0
                                         // 0 : デアサートを待つ
                                         // 3 : デアサートを待たない
    ULONG    ulCamPower;              // カメラ電源スイッチ(0:OFF、1:ON) ~ 初期値:1
    ULONG    ulI2CSpeed;              // I2C スピード(0:400Kbps、1:200Kbps、2:100Kbps) ~ 初期値:0
    ULONG    ulMonMode;               // リングバッファ数の指定 (2,8,12,16 から選択) ~ 初期値:2
    ULONG    ulPixellInfo;            // 画像情報通知bit0-1 : 色成分数(1or2or3)
                                         // bit2 : 1 色分のビット幅(0:8bit,1:16bit)
} SET_PARAM, *PSET_PARAM;
```

備考

VSYNC デアサートフラグの説明:

0 の場合、VSYNC (VD) がアクティブになったらフレームを認識します。

3 の場合、VSYNC (VD) がアクティブになる前に指定のサイズまで取り込んだらフレームを認識します。

モニタリングモードの説明:

SV ボード上の 256MB のメモリーを何分割するかを指定します。2 を設定すればダブルバッファとなります。設定は 2,8,12,16 の中から選択してください。それ以外は保証できません。

リング数が少ないとリアルタイム性が上がりますが、廃棄フレームの確立も高くなります。

リング数が多いと PC 側で負荷がかかった場合に SV ボード上のメモリーで一時的なバッファとなりますので、廃棄フレームの確立が低くなります。

モニタリング時の設定となり、レコーディングは無効となります。

画像情報通知の説明:

YUV-8 ビット取り込みの場合、色成分数は 2、ビット幅は 0 を指定します。

YUV-16 ビット取り込みの場合、色成分数は 2、ビット幅は 1 を指定します。

RAW-8 ビット取り込みの場合、色成分数は 1、ビット幅は 0 を指定します。

RAW-10 ビット取り込みの場合、色成分数は 1、ビット幅は 1 を指定します。

RGB-24 ビット取り込みの場合、色成分数は 3、ビット幅は 0 を指定します。

※ulVsyncFlag は未サポートとなります。

3.4.7. SVIUSB20_MonStart

API SVIUSB20_MonStart

機能 SV ボードへモニタリング開始を通知します

プロトタイプ

```
DWORD SVIUSB20_MonStart (
    PMONITOR_PARAM  pMonParam    // モニタリング情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct    _MONITOR_PARAM {
    ULONG    ulCutout_x;        // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG    ulCutout_y;        // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG    ulCutout_w;        // 切り出し範囲の幅(偶数)
    ULONG    ulCutout_h;        // 切り出し範囲の高さ
} MONITOR_PARAM, *PMONITOR_PARAM;
```

備考

・本 API の発行前に、SVIUSB20_GetStatus API を発行し、SVI アイドル状態であることを確認して下さい。

3.4.8. SVIUSB20_MonStop

API SVIUSB20_MonStop

機能 SVI へモニタリング停止を通知します

プロトタイプ

```
DWORD SVIUSB20_MonStop ( void );
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

・本 API の発行前に、SVIUSB20_GetStatus2 API を発行し、SVI 基本ステータスを取得して下さい。基本ステータスが SVI_STS_FRAMEPENDING (正常終了(保留フレーム/データあり)) の場合、保留フレームが SVI に残っているのでヘッダ、画像データを全部取り込んで下さい。

3.4.9. SVIUSB20_MonGetHeader

API SVIUSB20_MonGetHeader

機能 SV ボードよりモニタリングヘッダ情報を取得します

プロトタイプ

```
DWORD SVIUSB20_MonGetHeader (
    PMON_HEADER    pHeader    // ヘッダ情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _MON_HEADER {
    ULONG    ulStatus;                // 実行結果のステータス
                                           // (詳細は SVIUSB20_GetStatus の基本ステータスと同様)
    ULONG    ulPendingInfo;           // 保留フレーム数を返す
    ULONG    ulOrgSizeW;               // カメラが出力している画像サイズの幅
    ULONG    ulOrgSizeH;               // カメラが出力している画像サイズの高さ
    ULONG    ulCutout_x;               // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG    ulCutout_y;               // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG    ulMonSizeW;               // 切り出し後の画像サイズの幅
    ULONG    ulMonSizeH;               // 切り出し後の画像サイズの高さ
    ULONG    ulFrameRate;              // カメラが出力している画像のフレームレート
    ULONG    ulLoseFrame;              // 前フレームから本フレーム間に破棄したフレーム数
    ULONG    ulOpticalAxisInfo;        // 予約
    USHORT    usCursorX;               // 予約
    USHORT    usCursorY;               // 予約
    USHORT    usWakuLeftX;             // 予約
    USHORT    usWakuLeftY;             // 予約
    USHORT    usWakuRightX;            // 予約
    USHORT    usWakuRightY;            // 予約
    ULONG    ulReserve[1];             // 予約
    ULONG    ulVsyncReadLen;           // 予約
    ULONG    ulVsyncReadVal[448/4];    // 予約
} MON_HEADER, *PMON_HEADER;
```

備考

- 構造体メンバ ulPendingInfo (保留フレーム数) が 0 の時、既にモニタリング停止通知済みであれば全フレームを受信したことを表します。モニタリング停止未通知であれば何らかのエラーが発生しています。戻り値を確認して下さい。
- 構造体メンバ ulPendingInfo (保留フレーム数) が 1 または 2 の時、引き続き画像データの受信をして下さい。
- 画像データが YUV-8bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulMonSizeW} \times 2) \times \text{ulMonSizeH}$$
- 画像データが YUV-16bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulMonSizeW} \times 2) \times \text{ulMonSizeH}$$
- 画像データが RAW-8bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulMonSizeW} \times 1) \times \text{ulMonSizeH}$$
- 画像データが RAW-10bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulMonSizeW} \times 2) \times \text{ulMonSizeH}$$
- 画像データが RGB-24bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulMonSizeW} \times 4) \times \text{ulMonSizeH}$$

3.4.10. SVIUSB20_MonGetFrame

API SVIUSB20_MonGetFrame

機能 SV ボードよりモニタリング・フレームデータを受信します

プロトタイプ

```

DWORD SVIUSB20_MonGetFrame (
    LPVOID    lpFrameBuf,    // フレームデータ格納バッファのポインタ
    ULONG     ulRcvLen,      // 画像データのバイト数(64 の倍数)
    PULONG    pulRetRcvLen   // 実際に読み込んだバイト数を格納するポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります(ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

ulRcvLen と*pulRetRcvLen が異なるときは何らかのエラーが発生しています。

3.4.11. SVIUSB20_RecStart

API SVIUSB20_RecStart

機能 SV ボードにレコーディング開始を通知します

プロトタイプ

```

DWORD SVIUSB20_RecStart (
    PRECORD_PARAM    pRecParam    // レコーディング情報構造体のポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```

typedef struct _RECORD_PARAM {
    ULONG    ulFrameCount;    // レコーディングフレーム数
    ULONG    ulBufferCount;   // レコーディングサイズ(バイト)
    ULONG    ulRecMode;       // レコーディングモード
                                // bit31 : 0=Recording, 1=Monitoring&Save Mem
                                // bit30-01 : Reserve
                                // bit00 : 0=フレーム数, 1=サイズ
} RECORD_PARAM, *PRECORD_PARAM;

```

備考

- ・本 API の発行前に、SVIUSB20_GetStatus API を発行し、SVI がアイドル状態であることを確認して下さい。
- ・ulRecMode の Monitoring&Save Mem は未サポートとなります。

3.4.12. SVIUSB20_RecStop

API SVIUSB20_RecStop

機能 SV ボードにレコーディング停止を通知します

プロトタイプ

DWORD SVIUSB20_RecStop (void);

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

・本 API の発行前に、SVIUSB20_GetStatus2 API を発行し、SVI の基本ステータスを取得して下さい。基本ステータスが SVI_STS_FRAMEPENDING (正常終了 (保留フレーム/データあり)) の場合、保留レコーディング・データが SV ボード上に残っているのでヘッダ、画像データを全部取り込んで下さい。

3.4.13. SVIUSB20_RecGetHeader

API SVIUSB20_RecGetHeader

機能 SV ボードよりレコーディング・ヘッダ情報を取得します

プロトタイプ

```
DWORD SVIUSB20_RecGetHeader (
    PREC_HEADER pHeader // ヘッダ情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_PARAMETER

引数に間違いがあります

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _REC_HEADER {
    ULONG    ulStatus;                // 実行結果のステータス
                                           // (詳細は SVIUSB20_GetStatus の基本ステータスと同様)
    ULONG    ulPendingInfo;           // 保留レコーディング・データサイズを返す
    UBYTE    ubFirmVersion;           // ファームウェアバージョン番号
    UBYTE    ubHardVersion;           // ハードウェアバージョン番号
    USHORT   usNumChannel;            // 予約
    UBYTE    ubCompFlag;              // 予約
    ULONG    ulNumScan;               // データサイズ
    USHORT   usDataWidth;             // データ幅 (0:8bits、1:16bits、2:24bits)
    UBYTE    ubReserve[1];            // 予約
    ULONG    ulRecHeaderSize;         // レコーディングモードが Monitoring&Save Mem 時の
                                           // レコーディングフレームヘッダーサイズ
} REC_HEADER, *PREC_HEADER;
```

備考

・構造体メンバ ulPendingInfo (保留レコーディング・データサイズ) が実際に SV ボードが保持しているレコーディング・データサイズです。0 の場合は何らかのエラーが発生しています。ulStatus を確認して下さい。

3.4.14. SVIUSB20_RecGetFrames

API SVIUSB20_RecGetFrames

機能 SV ボードよりレコーディング・データを受信します

プロトタイプ

```

DWORD SVIUSB20_RecGetFrames (
LPVOID    lpFrameBuf    // レコーディング・データ格納バッファのポインタ
ULONG     ulRcvLen,      // 画像データのバイト数(64 の倍数)
PULONG    pulRetRcvLen   // 実際に読み込んだバイト数を格納するポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL 正常終了
SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります(ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN オープンされていません
その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

備考

ulRcvLen と*pulRetRcvLen が異なるときは何らかのエラーが発生しています。

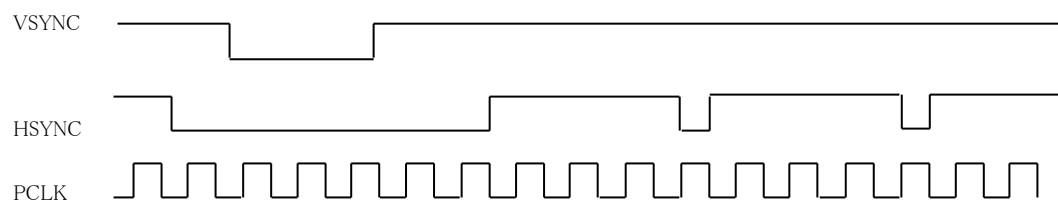
レコーディングの開始は、VSYNC の立ち上がりをトリガとしてデータの保存を開始します。
レコーディングで生成されるデータは映像取り込みビット数により 3 種類のフォーマットが存在します。

データ・フォーマット(レコーディングモードが Recording の場合)～映像取り込み 8/10/12bit の場合、
YUV-8bit、RGB565-8bit、RAW-8/10/12の映像を取り込んでいる場合、本APIにより取得したレコーディング・データは、1画素16ビットで取り込みビット数により、ビット格納に違いがあります。ビット8にVSYNCビット、ビット9にHSYNCビット、ビット0-7に映像データの低位8bitが格納されます。RAW-10/12の場合は下図の通りです。

ビット番号															
15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
---	---	D[11]	D[10]	D[9]	D[8]	HS	VS	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

YUV-8bit、RGB565-8bit、RAW-8ビット入力の場合、D[8]-D[11]は不定データとなります。
RAW-10ビット入力の場合、D8-D9が有効となり、D[8]-D[11]は不定データとなります。
RAW-12ビット入力の場合、D8-D11が有効となります。
ビット14-15は不定データとなります。

以下に概要例を示します。(YUV-8bit 入力時)



bit15-bit12	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
bit11-bit08	3	1	0	0	0	1	1	3	3	3	1	3	3	1	3
bit07-bit04	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit03-bit00	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

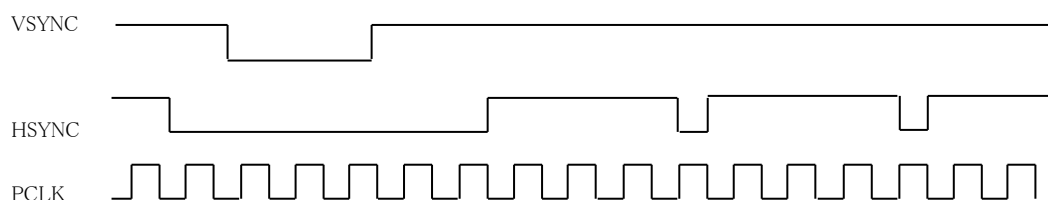
データ・フォーマット(レコーディングモードが Recording の場合)～映像取り込み 16bit の場合

YUV-16bitの映像を取り込んでいる場合、本APIにより取得したレコーディング・データは、1画素32ビットで下位16ビットに映像を記録し、ビット16にVSYNC、ビット17にHSYNCを記録します。その他のビットは不定データとなります。

ビット番号															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
31 (MSB)	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	—	—	—	—	—	—	—	—	—	—	—	—	HS	VS

YUV-16bitがビット0-15に記録され、ビット16にVSYNC、ビット17にHSYNCが記録され、その他は不定データとなります。

以下に概要例を示します。



bit31-bit28	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
bit27-bit24	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
bit23-bit20	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
bit19-bit16	3	1	0	0	0	1	1	3	3	3	1	3	3	3	1	3
bit15-bit12	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit11-bit08	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit07-bit04	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit03-bit00	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

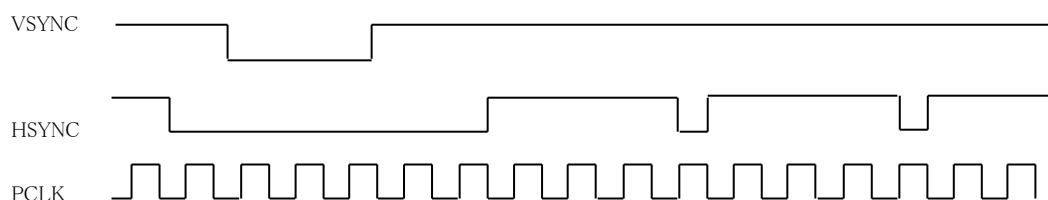
データ・フォーマット(レコーディングモードが Recording の場合)～映像取り込み 24bit の場合

RGB-24bitの映像を取り込んでいる場合、本APIにより取得したレコーディング・データは、1画素32ビットで下位24ビットに映像を記録し、ビット24にVSYNC、ビット25にHSYNCを記録します。その他のビットは不定データとなります。

ビット番号															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
31 (MSB)	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	—	—	—	—	HS	VS	D[23]	D[22]	D[21]	D[20]	D[19]	D[18]	D[17]	D[16]

RGB-24bitがビット0-23に記録され、ビット24にVSYNC、ビット25にHSYNCが記録され、その他は不定データとなります。

以下に概要例を示します。



bit31-bit28	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
bit27-bit24	3	1	0	0	0	1	1	3	3	3	1	3	3	3	1	3
bit23-bit20	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit19-bit16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit15-bit12	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit11-bit08	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit07-bit04	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit03-bit00	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

3.4.15. SVIUSB20_I2COneBlockWrite

API SVIUSB20_I2COneBlockWrite

機能 SV ボードに I²C で 1 ブロック送信します

プロトタイプ

```

DWORD SVIUSB20_I2COneBlockWrite (
    ULONG    ulSlaveAdr,      // I2C スレーブアドレス(7bit)
    ULONG    ulLen,          // 送信するコマンドデータ数
    ULONG    ulWriteMode,    // bit0-13 : デレイ時間 (0-10000, 単位 10usec で10の倍数で指定)
                                // bit14 : V 同期極性フラグ(0=立上がり, 1=立下り)
                                // bit15 : V 同期送信オンフラグ(0=OFF, 1=ON)
                                // bit16-30 : 予約(常に 0)
                                // bit31 : 再送オン (StopCondition 発行せず)
    PUCCHAR  pucSendBuf      // 送信コマンドデータバッファのポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H、bit23-0:基本ステータス)

備考

- デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。
- 本 API を発行することにより、スタートコンディション、データ送信、ストップコンディションを一回の転送で行うことができます。
- ulWriteMode のデレイ時間は V 同期送信オン状態で V 同期信号がアクティブになってからの時間を指定します。指定時間ウェイト後、I²C ブロックを送信します。再送オンは最後のストップコンディションを発行しません。
- スレーブアドレス 0x8 は SV ボードを指定します。

3.4.16. SVIUSB20_I2COneBlockRead

API SVIUSB20_I2COneBlockRead

機能 SV ボードより I²C で 1 ブロック受信します

プロトタイプ

```

DWORD SVIUSB20_I2COneBlockRead (
    ULONG    ulSlaveAdr,        // I2C スレーブアドレス(7bit)
    ULONG    ulLen,             // 読み出し要求バイト数
    ULONG    ulReadMode,        // bit0-13 : デレイ時間 (0-10000, 単位 10usec で10の倍数で指定)
                                // bit14 : V 同期極性フラグ(0=立上がり, 1=立下り)
                                // bit15 : V 同期送信オンフラグ(0=OFF, 1=ON)
                                // bit16-30 : 予約(常に 0)
                                // bit31 : 再送オン (Re-StartCondition 発行)
    PCHAR    pucRcvBuf          // 読み出しデータバッファのポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

- デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。
- 本 API を発行することにより、スタートコンディション、データ受信、ストップコンディションを一回の転送で行うことができます。
- 最後の1バイトのデータ受信時は ACK コードをスレーブデバイスに送信しません。
- ulReadMode のデレイ時間は V 同期送信オン状態で V 同期信号がアクティブになってからの時間を指定します。指定時間ウェイト後、I²C ブロックを送信します。再送オンは最後のストップコンディションを発行しません。
- スレーブアドレス 0x8 は SV ボードを指定します。

3.4.17. SVIUSB20_GpioAcc

API SVIUSB20_GpioAcc
 機能 SV ボードの GPIO ポート(入力 8 ポート、出力 8 ポート)をアクセスします
 プロトタイプ
 DWORD SVIUSB20_GpioAcc (
 ULONG ulMode, // 入出力フラグ(0:入力、1:出力)
 ULONG ulBitMask, // bit7 から bit0 でアクセスしたいポートのみ1をセット、0で無効
 PULONG pulData // 入出力データ(bit7 から bit0 が有効)
 ULONG ulRsv; // 予約
);

戻り値
 SVIUSB20_RET_NORMAL 正常終了
 SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります
 SVIUSB20_RET_ERROR_NOOPEN オープンされていません
 その他 Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)
 その他 デバイスエラー (bit31-24:F1H、bit23-0:基本ステータス)

備考

3.4.18. SVIUSB20_GetVersion

API SVIUSB20_GetVersion
 機能 SVIUSB20.DLL のバージョン情報を取得します
 プロトタイプ
 DWORD SVIUSB20_GetVersion(
 char *pcVerBuf // バージョン番号文字列を格納するポインタ
);

戻り値
 SVIUSB20_RET_NORMAL 正常終了
 SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります(ポインタがNULL)

備考

・以下のように文字列ポインタにバージョン番号が格納されます。

例) バージョン番号が 1. 0. 0. 0 の場合

```
* (pcVerBuf+0) = '1' // 0x31
* (pcVerBuf+1) = '.' // 0x2e
* (pcVerBuf+2) = '0' // 0x30
* (pcVerBuf+3) = '.' // 0x2e
* (pcVerBuf+4) = '0' // 0x30
* (pcVerBuf+5) = '.' // 0x2e
* (pcVerBuf+6) = '0' // 0x30
* (pcVerBuf+7) = '¥0' // 0x00
```

3.4.19. SVIUSB20_End

API SVIUSB20_End
機能 本ライブラリの終了処理を行います

プロトタイプ

```
void SVIUSB20_End( void );
```

戻り値

なし

備考

・必ず最後に呼び出して下さい。

3.4.20. SVIUSB20_DeviceSelect

API SVIUSB20_DeviceSelect
機能 使用するSVボードを指定します

プロトタイプ

```
void SVIUSB20_DeviceSelect (
    ULONG    ulBoardNo;    // SV ボード番号(0-4)
);
```

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_PARAMETER

引数に間違いがあります(ポインタがNULL)

備考

SV ボード番号はSVIUSB20_EnumDevice API で取得可能です。

3.4.21. SVIUSB20_DeviceRelease

API SVIUSB20_DeviceRelease
機能 使用指定したSVボードを開放します

プロトタイプ

```
void SVIUSB20_DeviceRelease ( );
```

戻り値

なし

備考

3.4.22. SVIUSB20_EnumDevice

API SVIUSB20_EnumDevice
機能 SV ボードの接続台数及び接続されたSV ボードの番号を接続台数分取得します
プロトタイプ

```
DWORD SVIUSB20_EnumDevice (
    ULONG    ulAppWho,          // オープン元のアプリケーションを指定
                                // SVIUSB20_APP_WHO_REC (表示アプリ用)
                                // SVIUSB20_APP_WHO_CTL (制御アプリ用)
    PULONG    *pulSVI_Num,      // 接続台数を格納するポインタ
    PULONG    *pulSVI_NumTable // 接続台数分のボード番号を格納する配列のポインタ
                                // このポインタがNULL の場合、接続台数のみ格納します
);
```

戻り値
SVIUSB20_RET_NORMAL 正常終了
SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります
その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

備考

- ・必ずSVIUSB20_Init API の後に呼び出して下さい。
- ・本API は最初に接続台数を検知するために呼び出し、接続台数分の配列(メモリ)を確保した後、再度呼び出しで各SV ボードのボード番号を取得できます。

3.4.23. SVIUSB20_RecStartEx

API SVIUSB20_RecStartEx
機能 リアルタイム版レコーディング開始を通知します (CAN 対応)

プロトタイプ

```
DWORD SVIUSB20_RecStartEx (
    PRECORD_PARAM_EX pRecParam // レコーディング情報構造体のポインタ
);
```

戻り値
SVIUSB20_RET_NORMAL 正常終了
SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN オープンされていません
その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _RECORD_PARAM {
    ULONG    ulRecMode;          // レコーディングモード
                                // bit31 : 0=CAN_0 not recording, 1= CAN_0 recording
                                // bit30 : 0=CAN_1 not recording, 1= CAN_1 recording
                                // bit29 : 画像データの詰め方指定 (0:LSB, 1:MSB)
                                // bit28-00 : 予約
} RECORD_PARAM_EX, *PRECORD_PARAM_EX;
```

備考

- ・本 API の発行前に、SVIUSB20_GetStatus API を発行し、SV ボードがアイドル状態であることを確認して下さい。
- ・本 API は SVI-09/SVM-06 CAN モードでサポートとなります。
- ・CAN_0 信号は GPIO12、CAN_1 信号は GPIO14 となります。
- ・ulRecMode で CAN 信号をレコーディングする、しない設定できますが、速度上の違いはありません。

3.4.24. SVIUSB20_RecGetData

API SVIUSB20_RecGetData

機能 リアルタイムにレコーディング・データを受信します (CAN 対応)

プロトタイプ

```

DWORD SVIUSB20_RecGetData(
LPVOID    lpFrameBuf    // レコーディング・データ格納バッファのポインタ
ULONG     ulRcvLen,      // 受信データのバイト数 (64MB を指定すること)
PULONG     pulRetRcvLen  // 実際に読み込んだバイト数を格納するポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります (ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

本レコーディングはリアルタイムにレコーディングデータを受信するために使用し、SVIUSB20_RecStartEX API と共に使用します。レコーディングの停止は SVIUSB20_RecStop API を使用します。

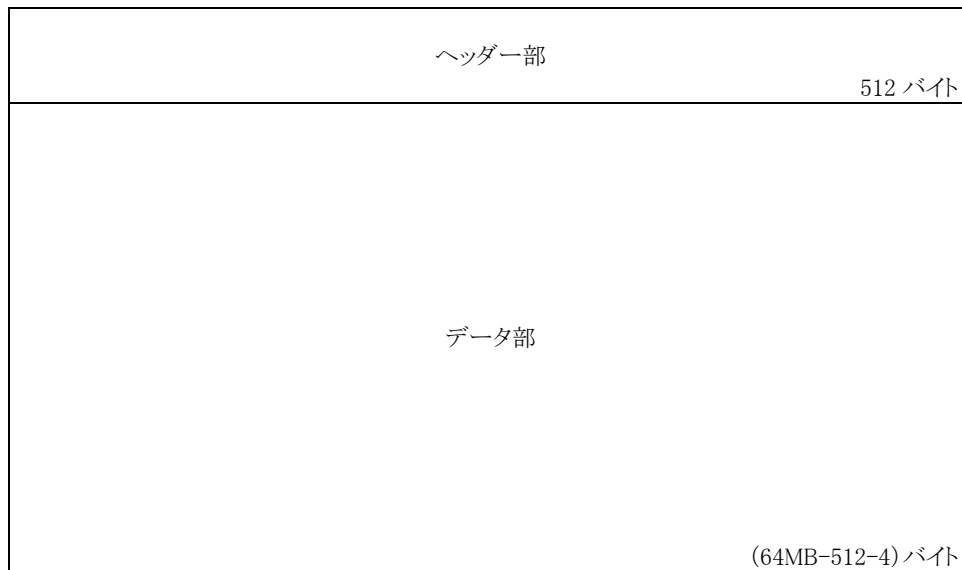
*pulRetRcvLen は ulRcvLen から 4 を引いた値が正解です、異なるときは何らかのエラーが発生しています。

ulRcvLen には 64MB (64 x 1024 x 1024) を指定してください。

受信データについては次ページをご覧ください。

データ・フォーマット全体

本 API により取得したデータは先頭に 512 バイトのヘッダー部と、続くデータ部で構成されます。



ヘッダー部

ヘッダー部は 512 バイト固定で下表の構成となっています。

Offset [DWORD]	Item	Bit Field	Description	Note
0	Status	[0]	常に'1'	
1	PendingInfo	[0]	次に読出し可能な Subframe がある場合に'1'	
2	NumCh HVerFVer	↓	(DRB では SmpClkSel のみ有効です。)	
		[7: 0]: FirmVer	(過去との互換、現状で 8'h00)	
		[15: 8]: HardVer	(過去との互換、現状で 8'h00)	
		[23:16]: NumChannel	(過去との互換、現状で 8'h00)	
3	NumScan CompFlagY	[26:24]: SmpClkSel	本サブフレーム取り込み時の DRB_SMPL_CLK_SEL レジスタ設定の値	
		↓		
		[7]: SLastFlag	一連のフレーム取り込みにおいて、最後のサブフレームかどうかを示します。	
4	DataWidth	[31:16]: SCntNum	サブフレームの番号を示します。一連のフレームの中で先頭のサブフレームは値'0'となります。	
		↓	サンプリング入力のデータ幅に関する設定を示します。	
		[23: 0]: BgmSygSel	本サブフレーム取り込み時の DRB_BGM_SYG_SEL レジスタ設定の値	
		[26:24]: BgpSygCnt	本サブフレーム取り込み時の DRB_BGP_SYG_CNT レジスタ設定の値	
5	HeaderSize	[31:28]: BgpError	[28]: DRB_EXC_BGP_SRC_OVERFLOW レジスタの値を示します。 [31]: DRB_BGP_EXC_FBP_LOGIC レジスタの値を示します。	
		[31: 0]	32'd512 固定	
6 ～ 15	<Reserved>	[31: 0]	常に 32'h0000_0000	
16	SubframeSize	[26: 2]	サブフレーム本体のサイズ、4Byte 単位	
17	SubframeTime	↓		
		[19: 0]: STime	サブフレームのタイム・スタンプ	

		[26:24]: STimeUnit	上記タイム・スタンプの時間単位設定を示します。本サブフレーム取り込み時の DRB_SHG_STIME_UNIT レジスタ設定の値	
		↓	サブフレーム・カウンタの状態について示します。	
18	SubframeCnt	[15: 0]: SCntNum	サブフレームの番号を示します。一連のフレームの中で先頭のサブフレームは値'0'となります。	
		[31:16]: SCntLose	直前に失われたサブフレームの数を示します。	
		↓	フレーム・カウンタの状態について示します。	
19	FrameCnt	[15: 0]: FCntNum	フレームの番号を示します。先頭のサブフレームは値'0'となります。	
		[23]: SLastFlag	一連のフレーム取り込みにおいて、最後のサブフレームかどうかを示します。	
		[26:24]: SmplClkSel	本サブフレーム取り込み時の DRB_SMPL_CLK_SEL レジスタ設定の値	
		↓		
20	FrameFmt	[23: 0]: BgmSygSel	本サブフレーム取り込み時の DRB_BGM_SYG_SEL レジスタ設定の値	
		[26:24]: BgpSygCnt	本サブフレーム取り込み時の DRB_BGP_SYG_CNT レジスタ設定の値	
		[31:28]: BgpError	[28]: DRB_EXC_BGP_SRC_OVERFLOW レジスタの値を示します。 [31]: DRB_BGP_EXC_FBP_LOGIC レジスタの値を示します。	
21	FrameSmplCnt	[31: 0]	一連のフレーム取り込みにおいてサンプリングしたデータの数を示します。途中のサブフレームの場合は、フレームの先頭からの累積数でおおよその値を示します。最後のサブフレームの場合には、フレーム全体での正確なサンプリング総数を示します。	※1
22	FrameTrgMode	[31: 0]	本サブフレーム取り込み時のトリガ・モードを示します。(現状はモード'0'のみに対応なので、32'h0000_0000 となります。)	
23 ～ 31	<Reserved>	[31: 0]	常に 32'h0000_0000	
32	FitPriComStts	[31: 0]	F-Info Table(Primary Side)の Common Status への参照	※2
		[31: 0] ↓		※2
33	FitPriEachCnts	[7: 0]:FRcnt	Frame 読出し側カウント値(新たに読出し可能な Frame の数)	
		[15: 8]:FWcnt	Frame 書込み側カウント値	
		[23:16]:FAcnt	Frame Table 上の実際のカウント値	
		[31:24]:FAcntMax	Frame Table 上で管理する最大数	
34 ～ 127	<Reserved>	[31: 0]	常に 32'h0000_0000	

※0. 全ての項目で明記されていない Bit Field については常に'0'となります。

※1. ただし、フレーム取り込み中にオーバー・フローが発生した場合は、欠落が発生しているので、サンプリング機関に対して正確なサンプル数とはなりません。

※2. Null-Frame-Header の場合に注記の項目に関してはヘッダ生成時点での有効な値となります。注記のない項目に関しては全て'0'となります。

※3. ヘッダ出力時に各項目の 32bit Field 毎に Big-Endian に変換します。上記は変換前の Little-Endian での表記です。

※4. FX3-I/F に対する転送設定により上記ヘッダの前に DSPH パケット・ヘッダが付加される場合があります。

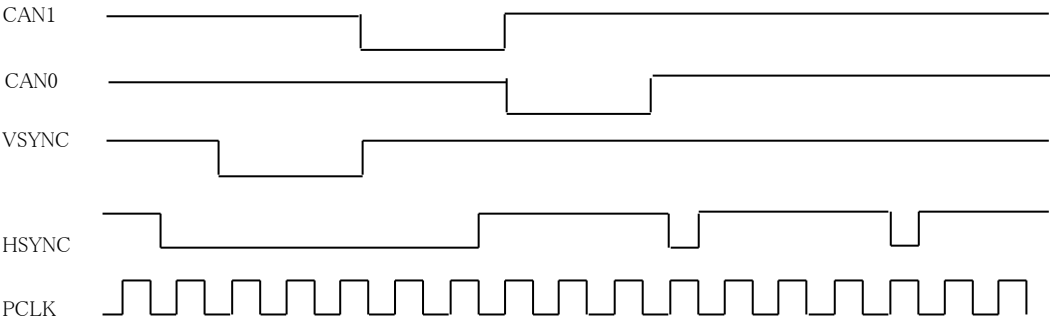
データ部

データ部は、1 画素 16 ビットで上位 8 ビットに 4 ビットの画像データと 2ch 分の CAN データ、VSYNC ビット(VS)、HSYNC ビット(HS) が格納され、下位 8 ビットに画像データが格納されます。

ビット番号															
15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
CAN1	CAN0	HS	VS	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

データの記録開始は、VSYNCの立ち下がりをトリガとしてデータの保存を開始します。フレームの区切りも同じポイントになります。
画像データは最大12bit記録できますが、8bit、10bitの場合はMSB詰めかLSB詰めかをSVIUSB20_RecStartEX APIの引数で指定します。

以下に概要例を示します。



bit15-bit12	F	D	C	C	C	5	5	B	B	B	D	F	F	F	D	F	F
bit11-bit08	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit07-bit04	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit03-bit00	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

3.4.25. SVIUSB20_RecStart2

API SVIUSB20_RecStart2

機能 SV ボードに開始トリガーなしのレコーディング開始を通知します

プロトタイプ

```
DWORD SVIUSB20_RecStart2 (
    PRECORD_PARAM    pRecParam    // レコーディング情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _RECORD_PARAM {
    ULONG    ulFrameCount;    // 未使用
    ULONG    ulBufferCount;    // ブロックサイズ(単位 MB)-当面 64 固定
    ULONG    ulRecMode;    // 未使用
} RECORD_PARAM, *PRECORD_PARAM;
```

備考

- 本 API の発行前に、SVIUSB20_GetStatus API を発行し、SVI がアイドル状態であることを確認して下さい。
- 本 API は SVI-09 / SVP-01-V のみサポートとなります。
- 本 API 発行後、次項の SVIUSB20_RecGetFrames2 API にてデータを取得してください。
- ブロックサイズで指定したサイズずつ SV ボードよりデータを取得します。
- データ取得終了時は必ず SVIUSB20_RecStop2 API を呼び出して下さい。

3.4.26. SVIUSB20_RecGetFrames2

API SVIUSB20_RecGetFrames2

機能 SV ボードより開始トリガ無しレコーディング・データを受信します

プロトタイプ

```
DWORD SVIUSB20_RecGetFrames2 (
    LPVOID    lpFrameBuf    // レコーディング・データ格納バッファのポインタ
    ULONG    ulRcvLen,    // 受信データのバイト数(単位 MB)-当面 64 固定
    PULONG    pulRetRcvLen    // 実際に読み込んだバイト数を格納するポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります (ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

- *pulRetRcvLen が 512 の時はデータが未受信状態を表します。
- *pulRetRcvLen が 512 ではなく、ulRcvLen と異なる場合は何らかのエラーが発生しています。
- 本 API 発行の前に、前項の SVIUSB20_RecStart2 API を呼び出して下さい。
- 受信したレコーディング・データの先頭 512 バイトはブロック・ヘッダが格納されており、ピクセルクロックに同期して 16bit 単位でレコーディングされます。
- データ取得終了時は必ず SVIUSB20_RecStop2 API を呼び出して下さい。

受信したデータフォーマット

ヘッダー部	512 バイト
データ部	(64MB-512-4) バイト

ヘッダー部は 512 バイト固定で下表の構成となっています。

Offset [DWORD]	Item	Bit Field	Description	Note
0	Status	[0]	常に'1'	
1	PendingInfo	[0]	次に読出し可能な Subframe がある場合に'1'	
2	NumCh HVerFVer	↓	(DRB では SmplClkSel のみ有効です。)	
		[7: 0]: FirmVer	(過去との互換、現状で 8'h00)	
		[15: 8]: HardVer	(過去との互換、現状で 8'h00)	
		[23:16]: NumChannel	(過去との互換、現状で 8'h00)	
		[26:24]: SmplClkSel	本サブフレーム取り込み時の DRB_SMPL_CLK_SEL レジスタ設定の値	
3	NumScan CompFlagY	↓		
		[7]: SLastFlag	一連のフレーム取り込みにおいて、最後のサブフレームかどうかを示します。	
		[31:16]: SCntNum	サブフレームの番号を示します。一連のフレームの中で先頭のサブフレームは値'0'となります。	
4	DataWidth	↓	サンプリング入力データ幅に関する設定を示します。	
		[23: 0]: BgmSygSel	本サブフレーム取り込み時の DRB_BGM_SYG_SEL レジスタ設定の値	
		[26:24]: BgpSygCnt	本サブフレーム取り込み時の DRB_BGP_SYG_CNT レジスタ設定の値	
		[31:28]: BgpError	[28]: DRB_EXC_BGP_SRC_OVERFLOW レジスタの値を示します。 [31]: DRB_BGP_EXC_FBP_LOGIC レジスタの値を示します。	
5	HeaderSize	[31: 0]	32' d512 固定	
6 ～ 15	<Reserved>	[31: 0]	常に 32'h0000_0000	
16	SubframeSize	[26: 2]	サブフレーム本体のサイズ、4Byte 単位	
17	SubframeTime	↓		

		[19: 0]: STime	サブフレームのタイム・スタンプ	
		[26:24]: STimeUnit	上記タイム・スタンプの時間単位設定を示します。本サブフレーム取り込み時の DRB_SHG_STIME_UNIT レジスタ設定の値	
		↓	サブフレーム・カウンタの状態について示します。	
18	SubframeCnt	[15: 0]: SCntNum	サブフレームの番号を示します。一連のフレームの中で先頭のサブフレームは値'0'となります。	
		[31:16]: SCntLose	直前に失われたサブフレームの数を示します。	
		↓	フレーム・カウンタの状態について示します。	
19	FrameCnt	[15: 0]: FCntNum	フレームの番号を示します。先頭のサブフレームは値'0'となります。	
		[23]: SLastFlag	一連のフレーム取り込みにおいて、最後のサブフレームかどうかを示します。	
		[26:24]: SmplClkSel	本サブフレーム取り込み時の DRB_SMPL_CLK_SEL レジスタ設定の値	
		↓		
20	FrameFmt	[23: 0]: BgmSygSel	本サブフレーム取り込み時の DRB_BGM_SYG_SEL レジスタ設定の値	
		[26:24]: BgpSygCnt	本サブフレーム取り込み時の DRB_BGP_SYG_CNT レジスタ設定の値	
		[31:28]: BgpError	[28]: DRB_EXC_BGP_SRC_OVERFLOW レジスタの値を示します。 [31]: DRB_BGP_EXC_FBP_LOGIC レジスタの値を示します。	
21	FrameSmplCnt	[31: 0]	一連のフレーム取り込みにおいてサンプリングしたデータの数を示します。途中のサブフレームの場合は、フレームの先頭からの累積数でおおよその値を示します。最後のサブフレームの場合には、フレーム全体での正確なサンプリング総数を示します。	※1
22	FrameTrgMode	[31: 0]	本サブフレーム取り込み時のトリガ・モードを示します。(現状はモード'0'のみに対応なので、32'h0000_0000 となります。)	
23 ～ 31	<Reserved>	[31: 0]	常に 32'h0000_0000	
32	FitPriComStts	[31: 0]	F-Info Table(Primary Side)の Common Status への参照	※2
		[31: 0] ↓		※2
33	FitPriEachCnts	[7: 0]:FRcnt	Frame 読出し側カウント値(新たに読出し可能な Frame の数)	
		[15: 8]:FWent	Frame 書込み側カウント値	
		[23:16]:FAcnt	Frame Table 上の実際のカウント値	
		[31:24]:FAcntMax	Frame Table 上で管理する最大数	
34 ～ 127	<Reserved>	[31: 0]	常に 32'h0000_0000	

※0. 全ての項目で明記されていない Bit Field については常に'0'となります。

※1. ただし、フレーム取り込み中にオーバー・フローが発生した場合は、欠落が発生しているので、サンプリング機関に対して正確なサンプル数とはなりません。

※2. Null-Frame-Header の場合に注記の項目に関してはヘッダ生成時点での有効な値となります。注記のない項目に関しては全て'0'となります。

※3. ヘッダ出力時に各項目の 32bit Field 毎に Big-Endian に変換します。上記は変換前の Little-Endian での表記です。

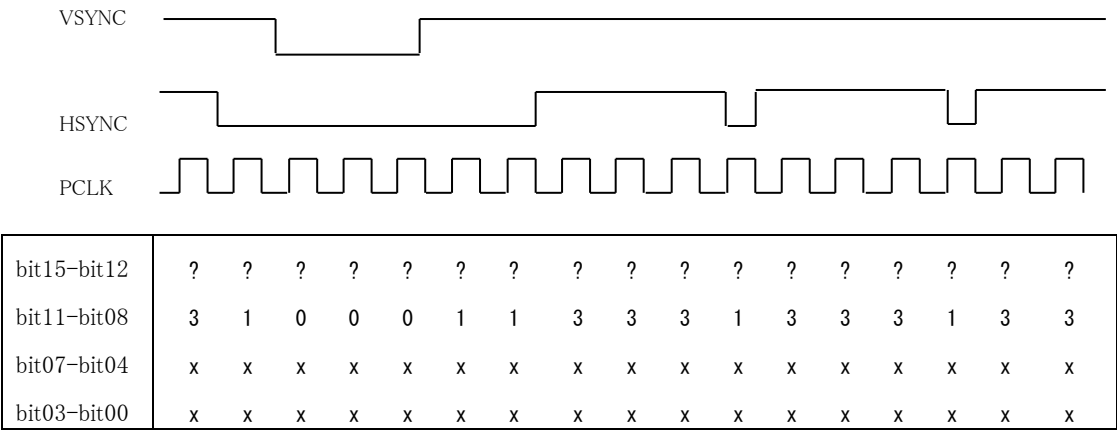
※4. FX3-I/F に対する転送設定により上記ヘッダの前に DSPH パケット・ヘッダが付加される場合があります。

データ部

データ部は、1 画素 16 ビットで上位 8 ビットに VSYNC ビット(VS)、HSYNC ビット(HS)が格納され、下位 8 ビットに画像データが格納されます。(ビット 10-15 は不定データとなります。)

ビット番号															
15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
---	---	---	---	---	---	HS	VS	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

以下に概要例を示します。



3.4.27. SVIUSB20_RecStop2

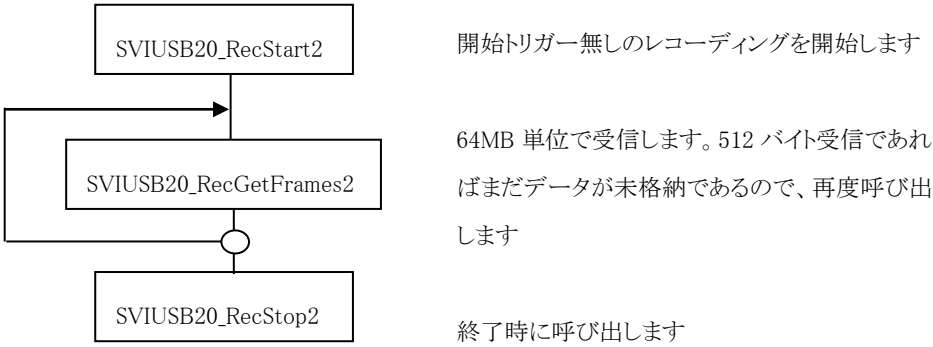
API SVIUSB20_RecStop2

機能 SV ボードに開始トリガ無しレコーディング停止を通知します

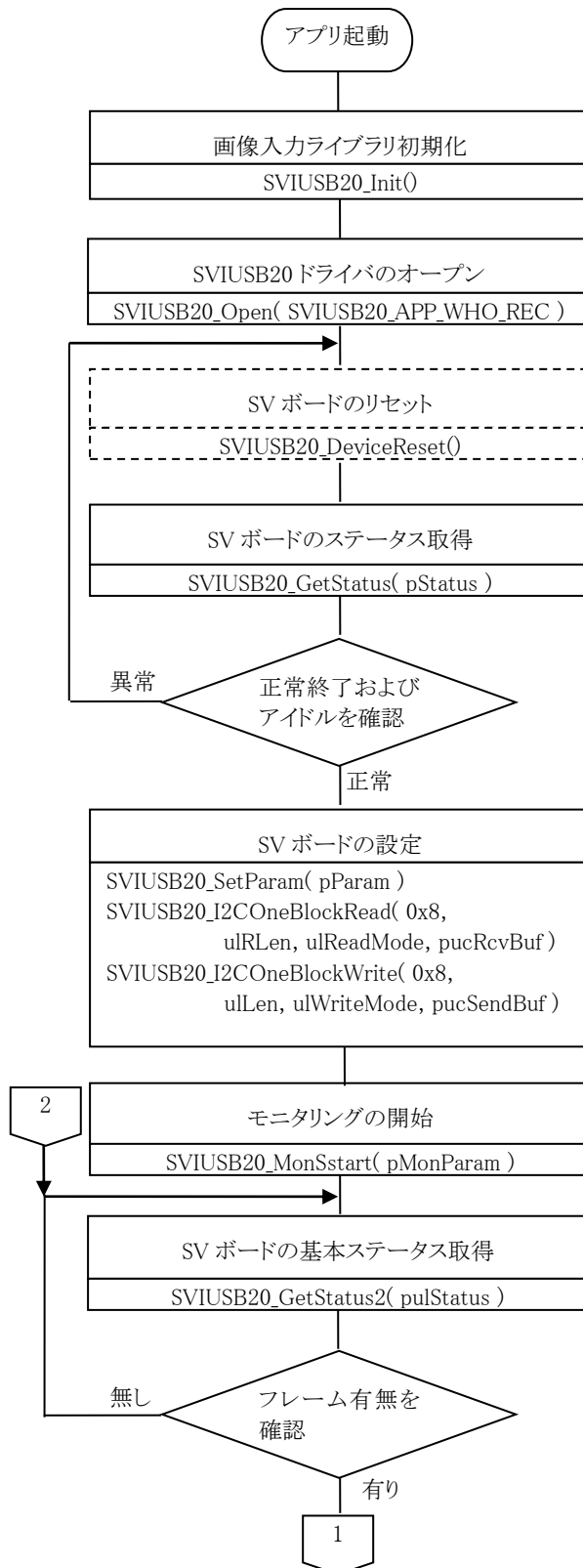
プロトタイプ
DWORD SVIUSB20_RecStop2 (void);

戻り値
SVIUSB20_RET_NORMAL 正常終了
SVIUSB20_RET_ERROR_NOOPEN オープンされていません
その他 Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考
・本 API は予め SVIUSB20_RecStart2 API を発行した動作に対して有効となります。



3.4.28. モニタリングモード時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要であれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

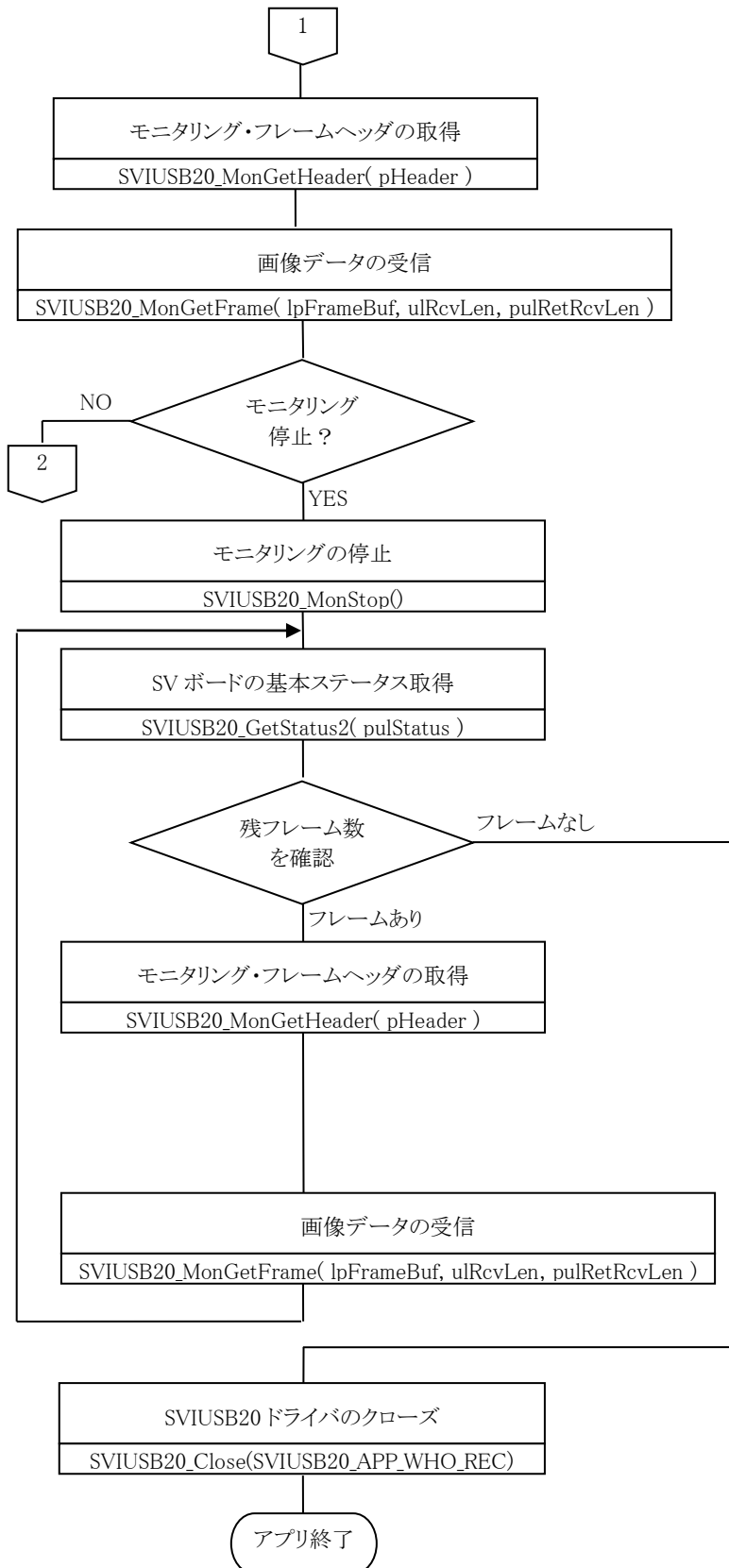
※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

詳細は「3.4.34.SV ボード設定」をご覧ください

※ SVIUSB20_MonSstart コール後、SVIUSB20_GetStatus2 で、SV ボードにキャプチャ画像フレームの有無を確認します。

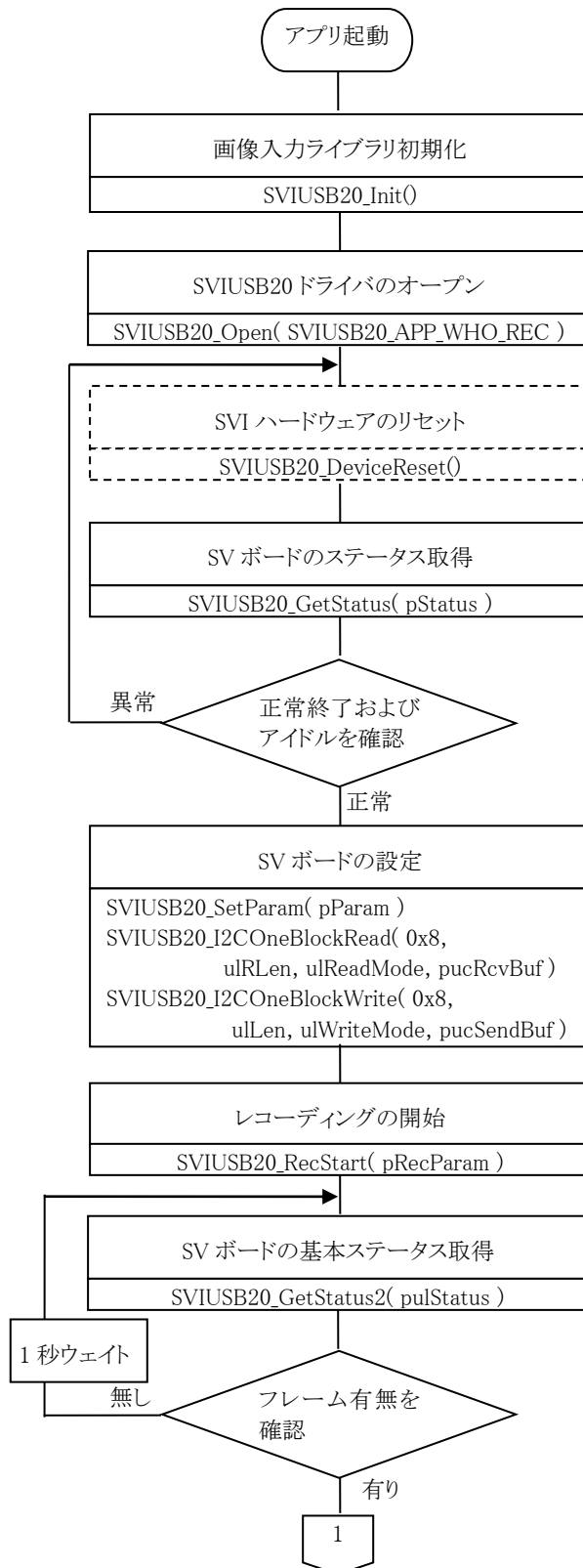


※ 取得したフレームヘッダを参照して
ulRcvDataSize を求めて下さい。

※ モニタリング停止でなければ連続して受信
して下さい。

※ 停止後、基本ステータスを取得し、残フレ
ームの有無を確認します。残フレームがなく
なるまで受信動作を繰り返します。

3.4.29. レコーディングモード時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

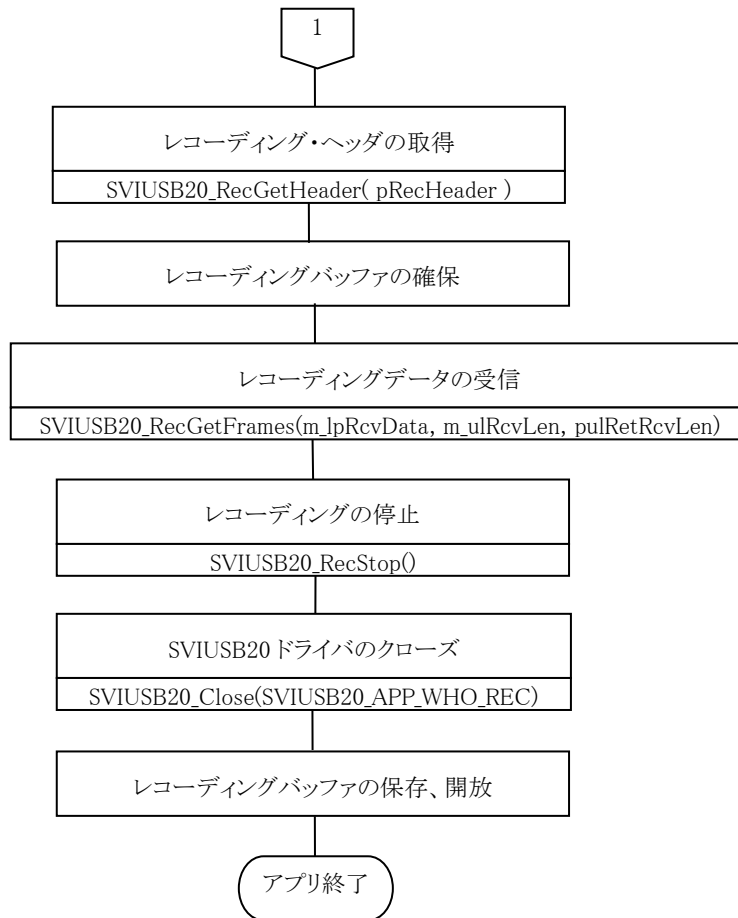
※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

詳細は「3.4.34.SV ボード設定」をご覧ください

※ SVIUSB20_RecStart コール後、SVIUSB20_GetStatus2 で、SV ボードにレコーディングデータの有無を確認します。

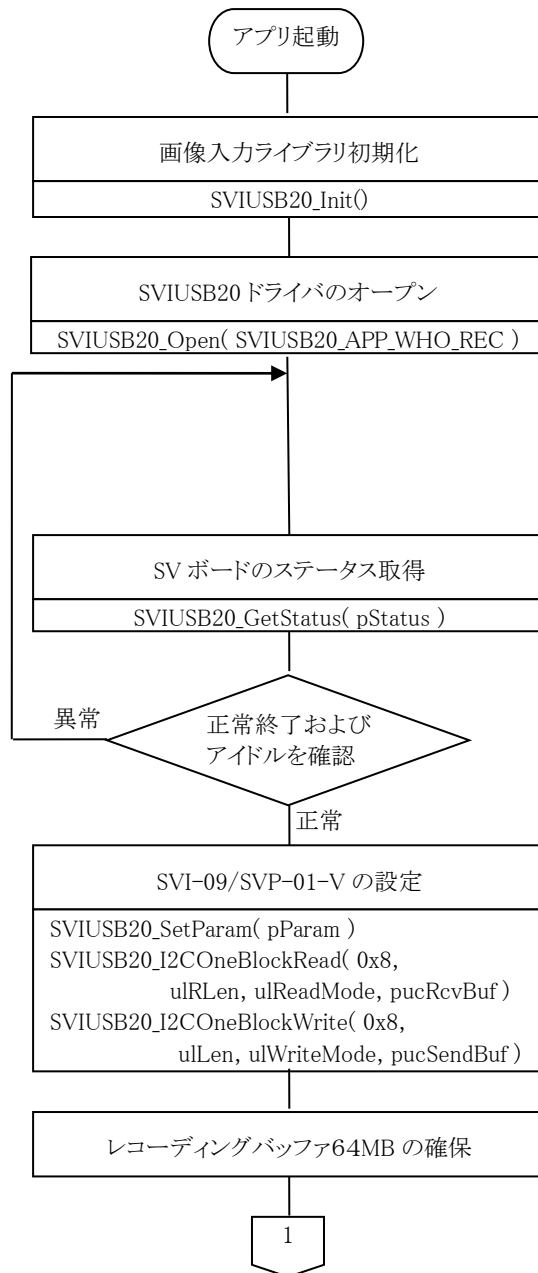


※ 取得したヘッダから RecHeader.ulNumScan がレコーディングした実バイト数です。この値を 64 で割り切れるよう桁上げた値が転送バイト数になります。

64の倍数に切り上げた値でメモリを確保してください。

m_lpRcvData は予め m_ulRcvLen 分確保しておいて下さい。

3.4.30. 開始トリガなしレコーディング時の画像入カライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

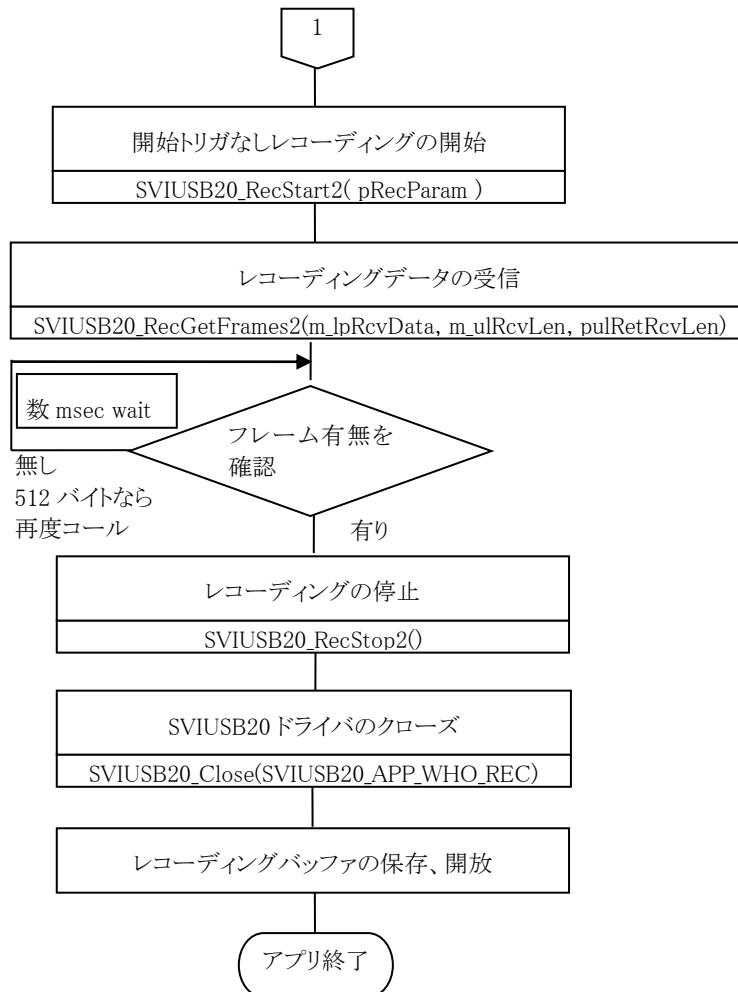
※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必要に応じて行って下さい。

詳細は「3.4.34.SV ボード設定」をご覧ください

※64MB 確保してください。

※PC 側の負荷を考慮して、64MB の領域を複数持って負荷を吸収させることも場合によっては必要になります。

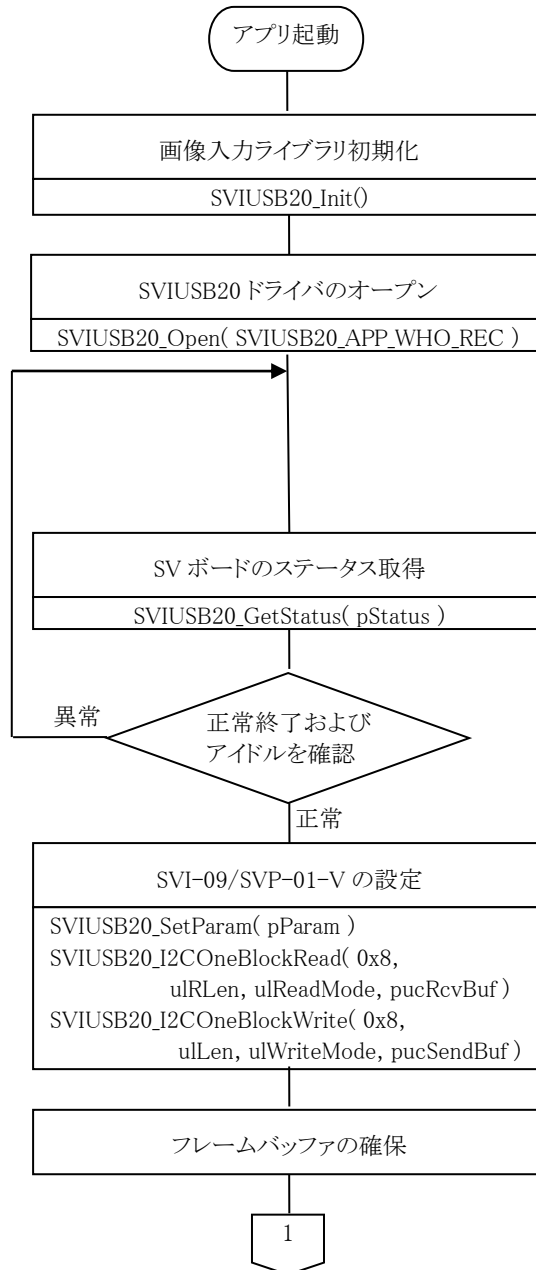


※SVIUSB20_RecStart2 コールします。

※受信データの戻りバイト数が 512 であればデータ部が格納されていないことになるので、数 msec ウェイト後、再度コールしてください。

※受信データをディスクへ抜け無しに保存する場合は SSD など高速なもの、受信と保存をスレッド分けるなど工夫してください。

3.4.31. JPEG など水平画素は一定でない場合のモニタリングの画像入カライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

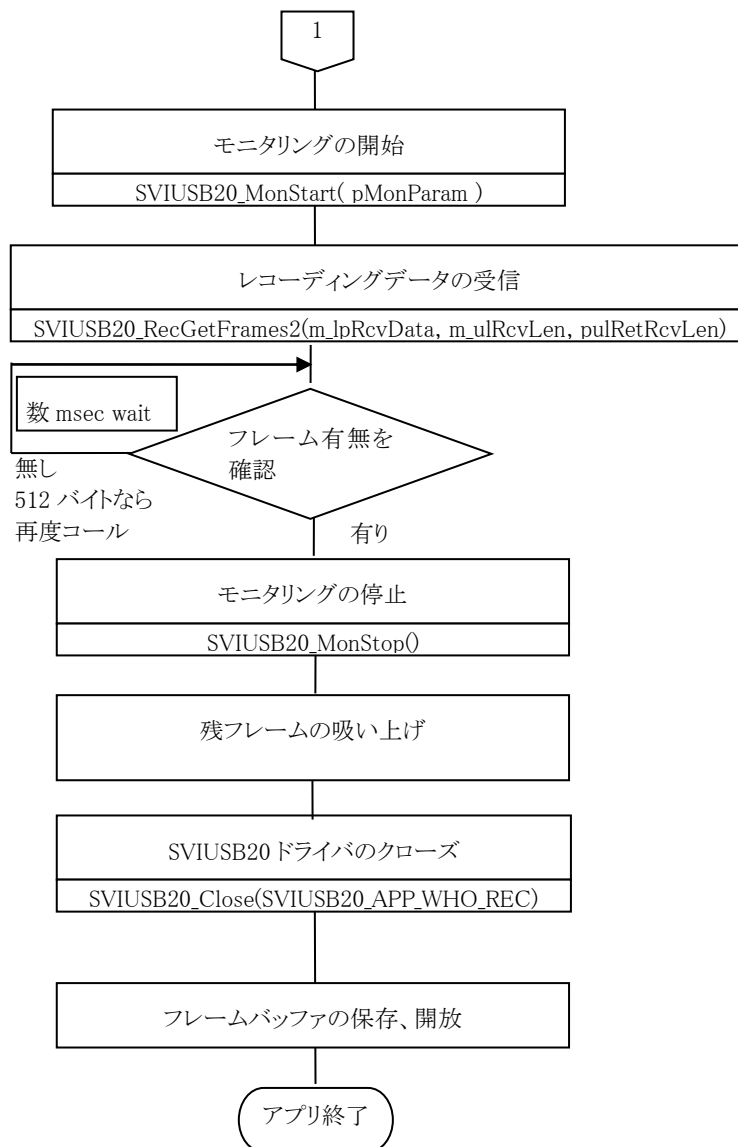
※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必要に応じて行って下さい。

詳細は「3.4.34.SV ボード設定」をご覧ください

※想定される1フレーム分の領域を確保してください。16MB 程度で妥当かと思います。

※PC 側の負荷を考慮して、領域を複数持つて負荷を吸収させることも場合によっては必要になります。



※SVIUSB20_MonStart コールします。

※受信データの戻りバイト数が 512 であればデータ部が格納されていないことになるので、数 msec ウェイト後、再度コールしてください。

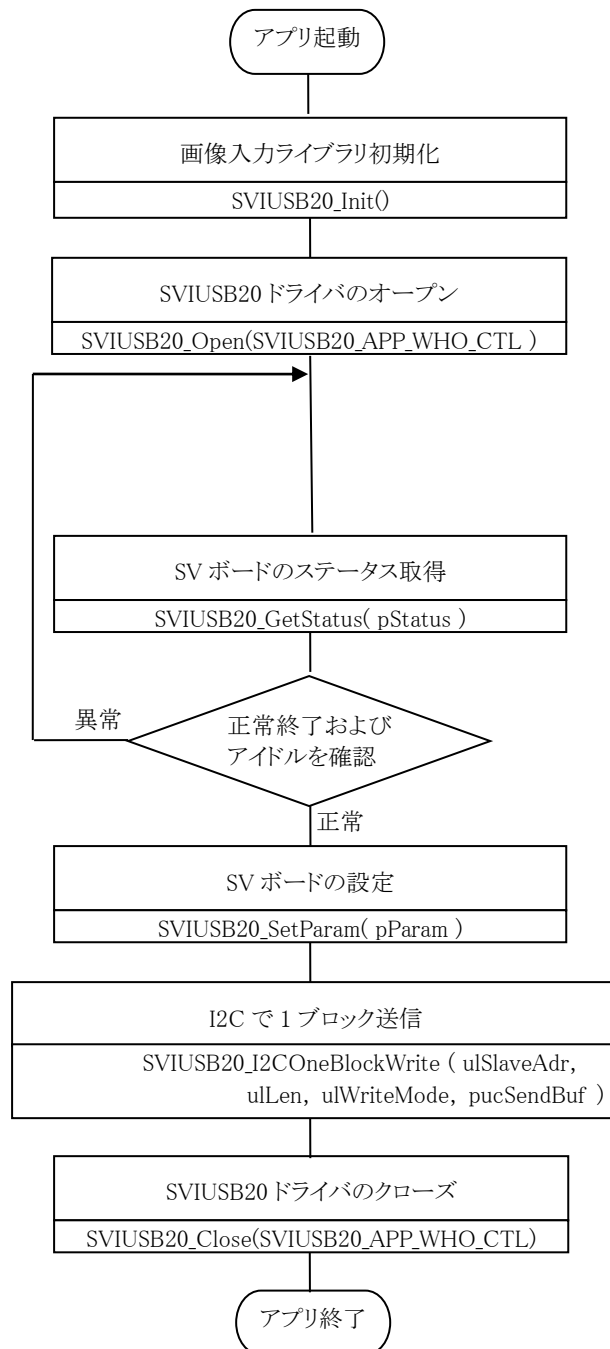
※ SVIUSB20_RecGetFrames2 の 前 に SVIUSB20_MonStart をコールしている場合は 1 フレームの生データを受信します。

※SVIUSB20_MonStart で開始したならば、SVIUSB20_MonStop で終わらせる必要があります。

※残フレームの吸い上げは、レコーディングデータ受信を何度か行い、数回データなし上であれば無くなったと認識して構いません。

※1 フレーム受信でディスクへ抜け無しに保存する場合は SSD など高速なもの、受信と保存をスレッド分けるなど工夫してください。

3.4.32. I2C によるコマンド送信時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時有効です。(3 秒かかります)

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

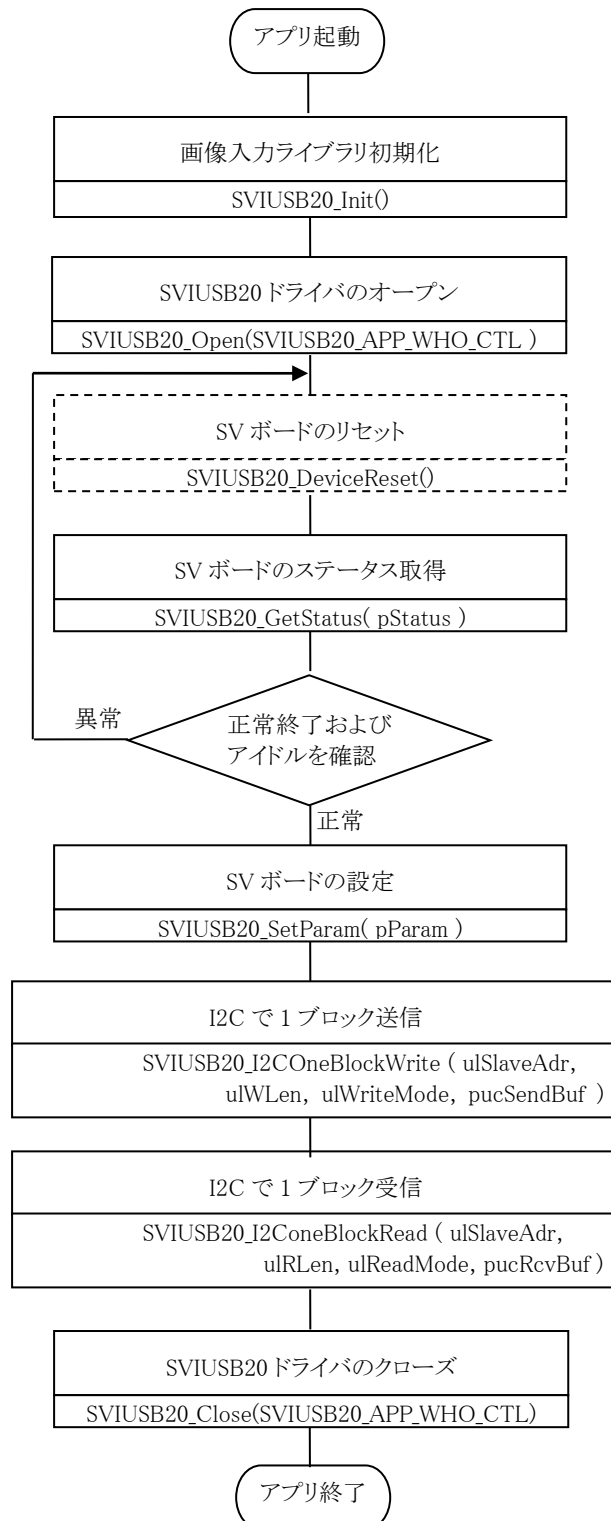
※必ず行って下さい。

※ulSlaveAdr にはスレーブ ID を代入する。

(API の中で左に 1 ビットシフトしている)

※ulLen にはスレーブ ID を含まないサブアドレスからのバイト数を指定する。

3.4.33. I2C によるコマンド受信時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

※ulSlaveAdr にはスレーブ ID を代入する。

(API の中で左に 1 ビットシフトしている)

※ulWLen にはサブアドレスのみなので 1 を代入する。

※ulRLen には受信するバイト数を指定する。

3.4.34. 複数台の SV ボードを使用する場合のオープン/クローズ方法

【オープン時】

①SVIUSB20_EnumDevice API を呼び出し、ライブラリーを初期化します。

②SVIUSB20_EnumDevice API を呼び出し、SVボードの接続台数を取得します。

```
DWORD dwRet;
ULONG ulSVI_Num;
dwRet = SVIUSB20_EnumDevice (
    SVIUSB20_APP_WHO_REC, // オープン元のアプリケーションを指定
                          // SVIUSB20_APP_WHO_REC (表示アプリ用)
                          // SVIUSB20_APP_WHO_CTL (制御アプリ用)
    &ulSVI_Num,           // 接続台数を格納するポインタ
    NULL                  // 接続台数分のボード番号を格納する配列のポインタ
                          // このポインタがNULL の場合、接続台数のみ格納します
);
```

③SVIUSB20_EnumDevice API を呼び出し、SVボードの接続台数分のボード番号を取得します。

```
PULONG pulSVI_NumTable;
pulSVI_NumTable = (PULONG) new ULONG[ulSVI_Num];
dwRet = SVIUSB20_EnumDevice (
    SVIUSB20_APP_WHO_REC, // オープン元のアプリケーションを指定
                          // SVIUSB20_APP_WHO_REC (表示アプリ用)
                          // SVIUSB20_APP_WHO_CTL (制御アプリ用)
    &ulSVI_Num,           // 接続台数を格納するポインタ
    pulSVI_NumTable       // 接続台数分のボード番号を格納する配列のポインタ
                          // このポインタがNULL の場合、接続台数のみ格納します
);
```

④ボード番号配列 (pulSVI_NumTable) から、SVボードのボード番号を抽出し、使用するSVボードを選択しします。選択はSVIUSB20_DeviceSelect API を呼び出します。

ボード番号配列pulSVI_NumTableのひとつの要素は上位16ビットにボード番号、下位16ビットにWindows管理番号を格納していますので、使用するボード番号が格納されている要素をSVIUSB20_DeviceSelect APIにて指定します。

```
SVIUSB20_DeviceSelect (
    pulSVI_NumTable[0]; // SVIボード番号を格納したデータ
);
```

⑤以降はSVIUSB20_Open APIにて使用できます。

【クローズ時】

①SVIUSB20_Close APIにてデバイスドライバをクローズした後、SVIUSB20_DeviceRelease APIを呼び出し、選択したSVIボードを開放します。

```
SVIUSB20_DeviceRelease ();
```

②SVIUSB20_End API を呼び出し、ライブラリを終了します。

```
SVIUSB20_End();
```

※同時接続ボード数ですが、4 台までです。

3.4.35. SV ボードの設定

モニタリング、レコーディングする際に、SV ボードの設定をしなければなりません。

設定は SVIUSB20_SetParam API、SVIUSB20_I2COneBlockWrite API、SVIUSB20_I2COneBlockRead API にて行います。

設定できる内容は、SVIUSB20_SetParam API の場合は、以下のとおりです。

画像情報通知	取り込みフレームの情報を通知します
カメラ電源スイッチ	モニタリング、レコーディングには関係ありません
I2C スピード	モニタリング、レコーディングには関係ありません
VSYNC デアサートフラグ	未使用
モニタリングモード	SV ボード上の SDRAM の使用方法を通知します

SVIUSB20_SetParam API については、「3.4.6. SVIUSB20_SetParam」をご覧ください。

SVIUSB20_I2COneBlockWrite API、SVIUSB20_I2COneBlockRead API にて設定できる内容は、別紙「SVI-09 I2C レジスタ表」に記載されているレジスタで

- ・取り込みデータビット幅の選択(アドレス 0h) ～ 8 ビットが初期値
- ・同期信号極性(アドレス 20h) ～ Low Enable が初期値
- ・データ取り込みタイミング設定(アドレス 2Eh) ～ DCK の立ち上がりでデータを取り込むが初期値となります。

例1) 取り込みデータビット幅を 16 ビットに変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x0; // サブアドレス 0h を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[1] );
ucBuf[1] |= 0x40;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 2, 0, &ucBuf[0] );
```

例 2) 同期信号極性を High Enable に変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x20; // サブアドレス 20h を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[1] );
ucBuf[1] |= 0x2;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 2, 0, &ucBuf[0] );
```

例 3) データ取り込みタイミングを立ち下がりに変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x2E; // サブアドレス 2Eh を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[1] );
ucBuf[1] |= 0x1;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 2, 0, &ucBuf[0] );
```

例 4) 同期信号極性(VSYNC)を入力段で反転する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x38; // サブアドレス 38h を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[1] );
ucBuf[1] |= 0x40; // VS Invert
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 2, 0, &ucBuf[0] );
```